一個相當於 BCNF 形式的 XML 文件正規化工具的實作

高國峰[1,2]　蔡儀男[1]　廖宜恩[1]

[1]國立中興大學資訊科學系、[2]修平技術學院資訊網路技術系

Email: kfkao@cs.nchu.edu.tw, mark@seed.net.tw , ieliao@nchu.edu.tw

摘　要

消除資料重複所導致的異常，是資料管理上非常重要的一個議題。在傳統的關聯式資料庫上，通常藉由正規化(Normalization)的過程，來消除資料異常的問題。在過去的研究當中，已經有學者將關聯式資料庫的概念延伸到 XML 文件當中。進而定義出了 XNF(XML Normal Form)。XNF 對資料的約束能力，被認為相當於關聯式資料庫的 BCNF。由於之前的研究多未實作。本論文進一步修改其演算法，並實作了一個能將 XML 文件，轉換成符合 XNF 的系統。我們的改善主要著重在清除修改觸發結構改變的問題。本論文所展示的工具，應是目前所知第一個具體實作的 XML 正規化工具。

關鍵詞 ：XML 、Schema 、Tree Tuples、XFD、XNF 。

1、 簡介

傳統的網頁，多以HTML(Hyper Text Markup Language)[10]語言撰寫而成。HTML標籤的功能，以描述資料呈現為主，並無法描述資料結構。為了能夠在網際網路的環境，有效的辨認及管理資料，XML[1,11]（eXtensible Markup Language）被提出來作為描述資料結構的一種共通標準。在XML的概念當中，所有的資料被組織成一個樹狀的結構，每一項資料並使用使用者自訂的標籤做明確的描述。由於XML本身是極具彈性的樹狀結構，允許每一個使用者自行定義樹狀組織的方式。所以一份資料，可能會有多種不同的結構來組織。因此，什麼樣的樹狀結構，會是一個好的樹狀結構，便成為一個很重要的課題。在此，好與不好的標準，是指該樹狀結構資料在更新時會不會出現資料不一致或者異常。會出現異常的樹狀結構，代表該份資料品質可能有問題。相對的設計良好的樹狀結構，可以避免資料不一致現象的產生。

在傳統的關聯式資料庫當中，資料品質的維護，是藉由正規化的程序來完成。經過正規化後的資料表，能夠保證資料的儲存不會重複，以確保不會有更新異常的產生。而依照不同的需求，學者提出了1NF、2NF、3NF、BCNF、4NF及5NF等不同層級的正規化，代表著對資料品質不同的要求。但是上述的正規化理論，是基於以集合為基礎的關聯式資料模型所定義出來的，因此並無法直接運用在XML的樹狀結構當中。

本篇論文的重點，便是希望找到一個XML資料正規化的有效方法。最早對XML正規化提出明確定義的論文，是Arenas 與 Libkin[2]在2002年發表的“A Normal Form for XML Documents”。在該論文中，作者描述了XML Functional Dependency(XDF)，以及XML Normal Form(XNF)的定義。作者並宣稱：XML資料模型的XNF，相當於關聯式資料模型的BCNF。該論文所發表的正規化演算法，也是現在許多XML相關研究的基礎與比較基準[3]。

以下列的資料courses為例， XML樹狀結構如圖1.1所示。在該資料中，每一個course元素會有一個課程編號（cno）屬性、課程名稱子元素以及修課學生的清單（taken_by）。修課學生的資料記錄在taken_by元素底下，每一個修課學生的資料包含學號（sno）屬性、姓名及成績子元素。該文件並遵守以下的條件限制：相同學號的學生，其姓名要一樣。我們也可將之描述為：學號決定姓名。換句話說，當學生的學號決定了，姓名也會跟著決定。上述件限制，將可以使用Arenas 與 Libkin所定義的XNF形式來表達。

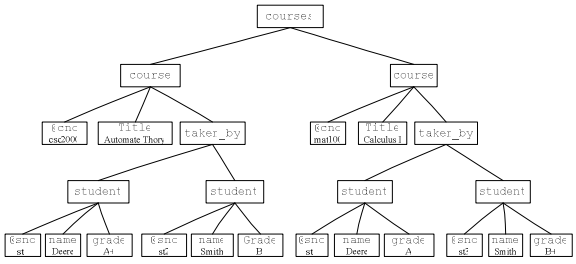

圖 1.1　courses XML Tree

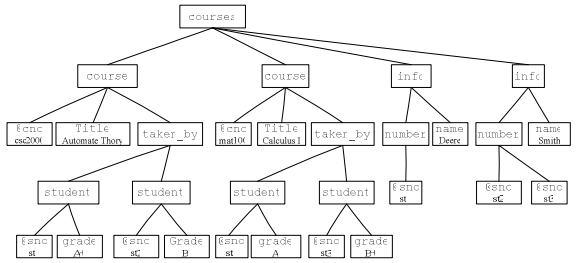

圖 1.2 courses 經過 Arenas 演算法正規化的 XML Tree

當該文件中，學生的姓名資料被更新時，就可能會產生不一致的問題。例如說：學號st1的學生要改名為Jone。因為該份資料儲存在兩個節點，若改名的動作只修改到一個節點，便會使該文件產生不一致的狀況。這種資料不一致的狀況，就是違反了上述關於學生姓名的條件限制。

為了避免資料不一致，依照Arenas 與 Libkin[2]

所發表的演算法，上述的文件將會被轉換成如圖1.2的文件，該結構已被證明是符合XNF的文件。圖1.2文件的缺點在於，修改現有資料時（例如說：學號st2的學生要改名為Smyth），可能仍需要新增節點，造成「修改」這個動作語意上的混淆。本研究，便是著眼於改善這些缺點，建立一個可實際執行的正規化系統。

2、 相關研究

XML文件正規化的研究，可以追朔到各類XML條件限制（constraint）的研究。XML文件所允許的結構，可以使用DTD或XML Schema[4,5]來表示。而每一個元素的內容是否恰當，則要仰賴各種的條件限制來維護。目前已被討論的有 Path constraints、Key constraints、Inverse constraints[6,7,8,9] 等等。在這些條件限制之中，最常被討論到就是Key constraints。而Key的概念，事實上，就是基於Functional Dependency（FD）的概念建上起來的。

一般來說，當XML文件符合所定義的constraint，且該constraint符合特定的規則，則稱該文件是符合XML Normal Form（XNF）。至於該規則如何定義，則不同的學者有不同的見解。

首先提出XNF並針對XNF下定義的是Embley及Mok在2001年發表的「Developing XML Documents with Guaranteed 'Good' Properties」[13]。在該論文當中，XNF必須符合特定的XML FD（XNF）、multivalued dependencies及inclusion constraints。在Embley及Mok的論文當中，上述的constraints及XML元素，都使用一個類似ER Model的視覺化方式來表示。Embley及Mok宣稱：符合其定義的XNF，將具有以下兩個特性。第一：XML內不會包含重複的資訊。第二：XML的schema tree將會是最小的。Embley及Mok所定義的XNF，包含了多種的constraint。所以相對於後續學者只使用XFD來定義XNF，並把XNF等同於RDB的BCNF，Embley及Mok的定義是比較嚴格的。但是在Embley及Mok所使用的類似ER Model的表示法當中，只偏重於表示nested relation的狀態，對於一般性的XFD並無法表達。

新加坡的學者Lee等人，在2002年於[12]這篇論文，提出表達XFD的語法。Lee等人提出的表示法為：$(p,[q_1,...,q_n \rightarrow q_{n+1}])$，其中 p 是XML中的path，且 $q_i = \tau.@l\ (i \in [1,n+1])$，其中 τ 為XML內的元素，l 為其屬性。這個表示法，類似RDB裏功能相依性的表示法。後續學者對XFD的表示方式，也多類似這樣的方式。

加拿大多倫多大學的Arenas及Libkin，在2004年發表了「A Normal Form for XML Documents」這篇論文。這篇論文，是第一篇以XFD定義XNF的論文，並証明XNF可以在資訊不會遺失（lossless information）的狀況底下，轉換成符合BCNF的關聯式資料表。當然，BCNF也被證明可以轉換成符合XNF的XML文件。由於該篇論文在研究上的創新及嚴謹性，使得該論文成為許多後續XML研究的基礎與比較對象。本論文的研究便是採用該論文語法，並延伸該論文的正規化演算法而加以改進。

緊接在Arenas及Libkin的論文之後，Vincent等也在2004年底發表了「Strong Functional Dependencies and Their Application to Normal Forms in XML」。這篇論文也是在定義XNF，討論過程中所使用的符號與語法也都跟Arenas及Libkin的論文類似。這兩篇論文對XNF定義的不同主要在於底下兩點。第一：Vincent等所發表的論文不使用DTD。第二：Vincent等的論文對NULL值，採取unk（unknown）的定義。並且在NULL值存在XML文件時，對XFD採取strong FD的定義。這如同RDB裏對incomplete relation，區分了strong Functional Depency與Weak Functional Depency兩種定義。在本論文中，僅在理論上對XNF的定義提出探討，並未更進一步提出不同的正規化演算法。

綜觀以上對XML正規化的研究，我們發現一個不符合XNF的XML檔，其實是可以轉化成許多種符合XNF的XML檔。但是，在眾多符合XNF的XML中，究竟那一種型式是比較符合使用者概念的XML，卻很少人討論。因此在本研究中，我們以Arenas及Libkin的演算法為基礎，提出新版本的正規化演算法。新演算法所轉換出來的XML，一樣符合Arenas及Libkin所定義的XNF，但包含了更多的考量，也將會更貼近使用者的概念。

3、 基本定義

本研究之基本定義採用 Arenas 及 Libkin[2]所給之定義。整理後，Arenas 及 Libkin 的定義描述如下：

定義 3.1（XML Schema Definition《XSD》）一個 XML 綱目定義被定義為 $D=(E,A,P,R,r)$，其中

$E \subseteq El$ 是一個有限的元素型態集合.

$A \subseteq Att$ 是一個有限的屬性集合.

P 是一個 Function 將 E 中的元素對應到元素型態定義.若是 τ 屬於 E，則 $P(\tau)$ = S 或是 $P(\tau)$ 是一個常規表示(regular expression) α：

α ::= ε | τ' | $\alpha|\alpha$ | α, α | $\alpha*$

其中 ε 代表空的元素集合，τ' 代表屬於 E 的元素，“|”、“,”和“*”分別代表 union、concatenation 和 Kleene closure。

R 是找出每一個在 E 中元素的屬性集合(powerset of A).

r 是XML Schema的根元素，不失一般性下，令 $r \neq P(\tau)$ 當 $\tau \in E$，符號 ε 表示元素型應是 EMPTY，S表示#PCDATA。

定義 3.2（路徑，path） XML Schema 的 path p 是一種路徑的描述方式，在 $D=(E,A,P,R,r)$ 中，若有一個節點的路徑 $l_1 \cdots l_n$，則 $l_i \in [2,n-1]$，若是 i= 1，則 l_1 為個 XML schema 樹的根節點。$l_n \in P(l_{n-1})$ 或是 l_n = @A 會屬於 $R(l_{n-1})$。$Paths(D)$ 表示 D(XML schema) 上的所有的 path 所形成的集合。$Epaths(D)$ 表示所有元素路徑(element path)的集合，也就是 $Epaths(D)$ 的 paths 不可以是以字串“S”或是屬性“@”為終點，以符號表示則 $Epaths(D) = \{ Paths(D) \mid last(p) \in E \}$，$last(p)$ 表示路徑 p 的終點元素。

定義 3.3（前序路徑，prefix path）假如p 是路徑$l_1 \cdots l_n$，q 是路徑$q_1 \cdots q_m$，若 p是q的 prefix path ，則說做p ⊆ q，其中n ≦ m ，$l_1 = q_1$，…，$l_n = q_n$，當 n = m 時，則 p = q 。如果p是q的prefix path ，但是 p ≠ q 則p為嚴格的前序路徑(strict prefix path)。

定義 3.4 (XML Tree) 一個XML Tree T 為一棵樹，用符號來表示則為 ：*(V, lab, ele, att, root)* 其中

V 是一個有限的點集合。

lab 是一個函數: $V \rightarrow E1$，將 *V* 中的點(node)對應到 *E* 中的元素。

ele 是一個函數: $V \rightarrow Str \cup V^*$ 找出 V 中的點的子節點。

att 是一個 partial function : $V \times Att \rightarrow Str$ 。

Root 是 T 的根節點。

定義 3.5 (tree tuples) 給一個XML Schema *D=(E, A, P, R, r)*，*D* 裏頭的每一個 tree tuple t都是一個函數t，t:$paths(D) \rightarrow Vert \cup Str \cup \{\perp\}$使得 ：

1. 若p ∈ *Epath(D)*，則 $t(p) \in Vert \cup \{\perp\}$，但 $t(r) \neq \perp$(NULL).
2. 若 p ∈ *Path(D)* – *Epath(D)*，則 $t(p) \in Str \cup \{\perp\}$.
3. 若 $t(p_1) = t(p_2)$ 而且 $t(p_1) \in Vert$ ，則 $p_1 = p_2$.
4. 若 $t(p_1) = \perp$ 而且 p_1 是 p_2 的 *prefix path* ，$t(p_2) = \perp$.
5. 若{ p ∈ *Path(D)* | $t(p) \neq \perp$}，則 *p* 會是一個有限的路徑.

定義 3.6 (*tuples*$_D$) 存在一個XML Schema *D*和一個 XML Tree *T*，且*T*遵守*D*結構，$tuples_D(T)$是T所有Tree Tuples t的值域化做tuple集合。

定義 3.7(XFD functional dependencies for XML) 在任何一個XML Schema *D*上，我們使用tree tuples來定義XFD，而*functional dependency (FD)*是指在*D*上具有*S*1 → *S*2形式的一種的相依性，這裏的*S*1、 *S*2是paths(*D*)的有限的、非空的子集合，所有在D上的FDs用FD(*D*)。

若有$S \subseteq paths(D)$,和有一個 $t, t \in T(D), t.S = t'.S$ 意味著 $t.p = t'.p$ 則所有的$p \in S$。$t.S \neq \perp$意味著$t.p \neq \perp$則所有的$p \in S$。

如果 $S1 \rightarrow S2 \in FD(D)$ ，*T*為一個 XML tree 而且 *T* 遵守 *D* 和$S1 \cup S2 \subseteq paths(T)$，則我們可以說 *T*是 滿足$S1 \rightarrow S2$ (可以 "$T \models S1 \rightarrow S2$"表示)。如果$t1, t2 \in tuples_D(T)$，$t1.S1 = t2.S1$ 且 $t1.S1 \neq \perp$則$t1.S2 = t2.S2$。 則我們觀察到上述的 *tree tuple* t1、t2是滿足FD $S1 \rightarrow S2$。

然而如有一path $p \in S2$, $t1.p$ and $t2.p$中有一個為null值或是二者皆為非unll值，而且有一個 在XML tree *T*的二個tree tuples *t*1, *t*2 有一關係 $t1.S1 = t2.S1$ 則意味著在某些$p \in S1$中會有null值。

在等價關係中，如有任有一FD φ，和 $T \equiv T'$， $T \models \varphi$則 $T' \models \varphi$. 我們可以寫成 $T \models \Sigma$且$\Sigma \subseteq FD(D)$。

如果 $T \models \varphi$ 而且每一個$\varphi \in \Sigma$，則以 $T \models (D, \Sigma)$, 如$T \models D$ 和 $T \models \Sigma$表示。

定義 3.8 $(D, \Sigma)^+$

若有一個 XML Schema *D*，並存在一個集合$\Sigma \subseteq FD(D)$ 以及一個 $\varphi \in FD(D)$。若任何遵守(D, Σ)的 XML 文件，也必然會遵守 φ；哪我們可以說：(D, Σ)可推導出 φ。所有可從(D, Σ)推導衍生出的 FD 集合，用$(D, \Sigma)^+$表示。

定義 3.9(XNF: AN XML NORMAL FORM)

如果有一個 XML Schema *D*和$\Sigma \subseteq FD(D)$，而且(D, Σ)是存在 XML normal form(XNF)，假設 S → *P.@l* 或 S → *P.S*是存在$(D, \Sigma)^+$裏的 nontrivial FD 則會有 S → *P*在$(D, \Sigma)^+$。

所以在直覺下，如 *S* → *P.@l* 是存在$(D, \Sigma)^+$裏。如果有 *T* 為一個 XML tree 而且 *T* 遵守 *D* 和滿足 Σ，然後在 *T* 中的 *S* 所有元素的值集合，我們可發現 p.*@l* 都有唯一的值，因此 *S* 所有元素的值集合需要存放 *p.@l* 的值只可一次，換句話說 *S* → *P* 必須包含於$(D, \Sigma)^+$中。

4、 修正 Arenas 及 Libkin 的正規化演算法

Arenas 及 Libkin 的正規化演算法所產生的 XML 文件，已經被證明是符合 XNF。然而，這個演算法所產生文件仍舊有缺點。這些狀況是發生在，處理重複值產生的異常 XFD 時產生的。我們歸納其缺點如下：第一：修改現有資料時，可能需要新增節點，造成「修改」這個動作語意上的混淆，同時也會增加維護上的複雜度。第二：過於瑣碎的正規化步驟，將造成 XML 文件結構上的一再重複。以下將分別討論產生這些缺點的原因，以及我們的處裡策略。

4.1 清除修改觸發結構改變的問題(XNF1)

第一個缺點之所以產生的原因，是當兩個不同的決定元素，具有相同的被決定元素值時，我們將會將它們放在一起（假設此異常 XFD 為 A→B，我們稱 A 為決定元素，B 為被決定元素）。例如說第一章圖 1.1 中，student 元素底下的@sno 跟 name 這兩個元素，@sno 將決定 name。而因為第一個 student 跟第三個 student 擁有相同的 name，所以轉換後的 XML 文件將它們擺在一起，如圖 1.2 的第二個 info 所示。所以當這兩組資料其中一組的被決定值修改，而導致它們不再相同時，便需要再將它們拆開。拆開之後，會先比較修改後的值，是否與其他 student 的 name 值相同。若相同，會將它們又歸納在一起；若不同，則需要新增一個 info 節點來儲存資料。這將導致修改的語意，卻需要藉由新增節點的動作來完成。

處理這個問題的策略便是讓 A 與 B 元素，在轉換過後的 XML tree 裡，仍保持每組資料一一對應的狀

態，如圖 4.1 所示。如此，雖然會新增一些節點來儲存資料，但整個語意將會較為清晰，修改的動作也不再需要比對其它舊值得運算。

為了和 Arenas 與 Libkin 所提出的演算法 XNF 有所區分，我們將清除修改觸發結構改變的問題(Eliminating the problem of structure changes triggered by modification)演算法定為 XNF1 演算法。

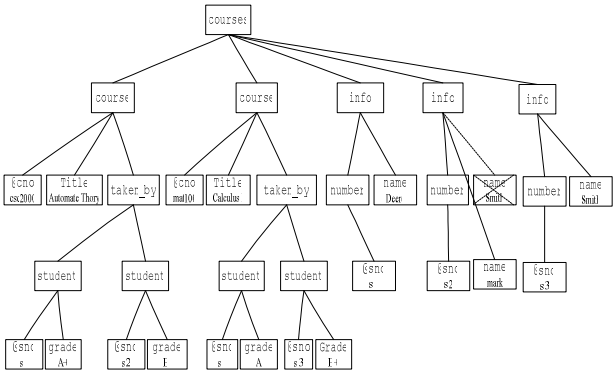

圖 4.1　courses.XML 以 XNF1 演算法正規化後

4.2 應用聯集推論法則清除重複結構問題(XNF2)

第二個缺點之所以產生的原因，在於當同時存在多條具有相同決定元素的異常 XFD 時，Arenas 及 Libkin 的演算法將會一條一條的去處理，如此過於瑣碎的正規化步驟，將造成 XML 文件結構上的一再重複。以第一章圖 4.3 的 course2 為例。因為同時存在學號決定姓名、學號決定 email、學號決定 email 等異常 XFD。依照 Arenas 及 Libkin 的正規化演算法，轉換過後的 XML 文件將使用三個 info 節點，來表示一個學生的姓名、email 及地址資料（圖 4.4），如此將造成結構上的重複。

為了處理這個問題的策略，本演算法運用了 1974 年 Armstrong 提出阿姆斯壯推論規則(Armstrong's Inference Rule)中的聯集規則:

Union rule(聯集性)：如果 A→B 且 A→C，則 A→BC。

表示式 :{ A→B 、A→C} ⊨ A→BC

修正 Arenas 及 Libkin 便是讓多條具有相同決定元素的異常 XFD，同時處理。例如說：異常 XFD 有學號→姓名、學號→email 及學號→地址，我們先將之合併成學號→{姓名、email、地址}，姓名、email、地址將被一並處裡一併移動。依此方法，圖 4.1 的 course2 Tree、圖 4.2 course2 XML Schema 圖將被轉換成如圖 4.5 所示。

為了和 Arenas 與 Libkin 所提出的演算法 XNF 有所區分，我們將應用聯集推論法則清除重複結構問題(Eliminating duplicated structures by using union inference rule)的演算法定為 XNF2 演算法。

修改後演算法所產生的 XML 文件，不會產生新的異常 XFD。因此，XML 文件也將會符合 XNF。上述的演算法，使用第三節介紹的符號，可以表示如下：

$D = (E, A, P, R, r)$是一個 XML Schema，存在一個集合 $\Sigma \subseteq FDs(D)$，假設(D, Σ)包含一個 anomalous FD $\{q, p_1.@l_1, \cdots, p_n.@l_n\} \rightarrow \{z_1.@J_1, \cdots, z_m.@J_m\}$，而且 $q \in EPaths(D).and\ n > 1$、 m>1 。

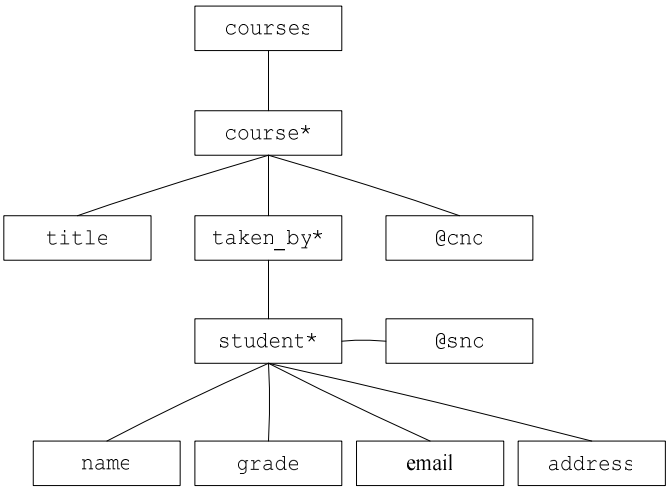

圖 4.2　courses2 XML Schema Tree

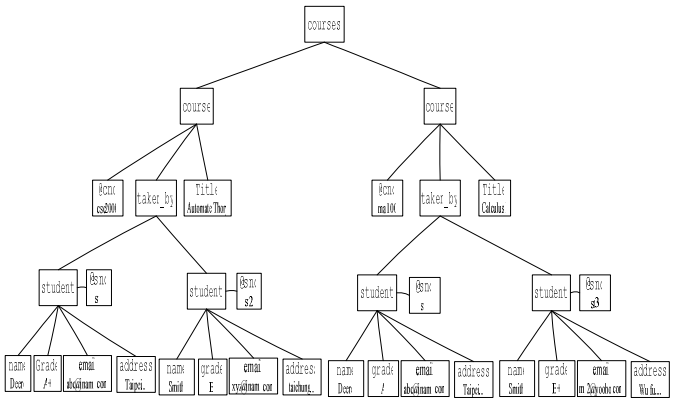

圖 4.3 courses2.XML Tree

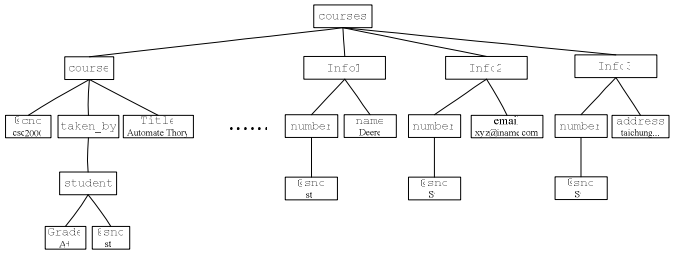

圖 4.4courses2.XML 經過 Arenas 演算法正規化後

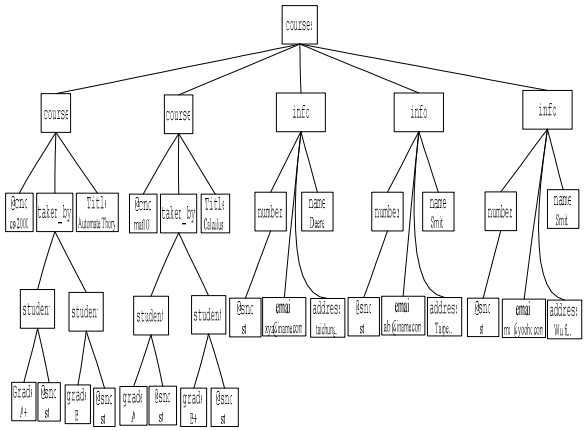

圖 4.5　courses2.XML 以 XNF2 演算法正規化後

針對 anomalous FD，Arenas 及 Libkin 所提的轉

換步驟，如圖 4.6 XML Schema 轉換示意圖所表示。先在 $last(q)$建立一個 element τ的子節點，並在 τ下建立 n 個子節點，分別移動$@l_1, \ldots, @l_n$到 $\tau_1, \ldots, \tau_n$ 屬性集合裏，再分別移動 p 下的$@l$、z_l下的$@J_l$到 τ,..., z_m下的$@J_m$到 τ，再把$@l$、$@J_1, \cdots, @J_m$刪除。

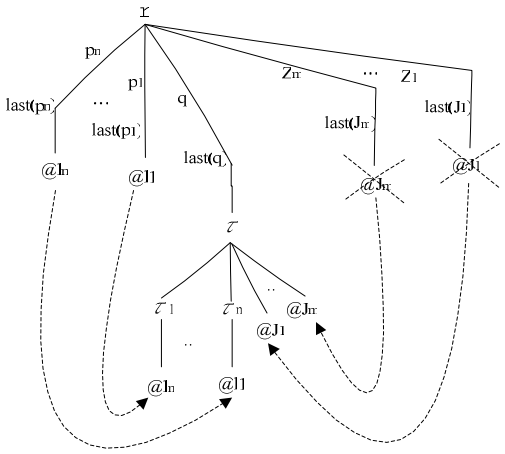

圖 4.6 XML Schema 轉換示意圖

令新的XML Schema $D[[p.@l, z_l.@J_l, \cdots, Z_m.@J_m] := q.\tau[\tau_1.@l_1, \ldots, \tau_n.@l_n]]$，$[@l, @J_l, \cdots, @J_m]$是一個屬性用$(E', A', P', R', r)$來表示，$E' = E \cup \{\tau, \tau_1, \ldots, \tau_n\}$。

1. 如果$P(last(q))$是一個regular expression ，用s表示之

則$P'(last(q)) = (s, \tau^*)$

$P'(\tau) = (\tau_1^*, \ldots, \tau_n^*) \circ P'(\tau_i) = \varepsilon$ 而且 $i \in [1, n]$

$P'(\tau') = P(\tau')$ 而且 $\tau' \in E - \{last(q)\}$

2. $R'(\tau) = \{@l, @J_l, \cdots, @J_m\}$，$R'(\tau_l) =$而且 $i \in [1, n]$

$R'(last(p)) = R(last(p)) - \{@l\}$ and $R'(\tau') = R(\tau')$ for each $\tau' \in E - last(p)$。

將D做轉換後，得到新的XML Schema $D' = [[p.@l, z_l.@J_l, \cdots, Z_m.@J_m] := q.\tau[\tau_1.@l_1, \ldots, \tau_n.@l_n]]$，由於轉換的動作已經改變了XML Schema的結構，所以連帶的產生了新的FD，$\Sigma' = \Sigma[[p.@l, , z_l.@J_l, \cdots, Z_m.@J_m] := q.\tau[\tau_1.@l_1, \ldots, \tau_n.@l_n]]$，$\Sigma'$包含$\Sigma$裏所有的FD，是由(1)、(2)和(3)構成的。

1. $S_l \rightarrow S_2 \in (D, \Sigma)^+$ 且 $S_l \cup S_2 \subseteq paths(D')$

$P'(\tau) = (\tau_1^*, \ldots, \tau_n^*) \circ P'(\tau_i) = \varepsilon$ 而且 $i \in [1, n]$

$P'(\tau') = P(\tau')$ 而且 $\tau' \in E - \{last(q)\}$

2. 當$p_i.@l_i (i \in [1, n])$ 和$p.@l$、$z.@z$被移到 τ和他的子節點，如果 $S_l \cup S_2 \subseteq$ $\{q, p_l, \cdots p_n, p_l.@l_1, \cdots p_n.@l_n, p.@l@J_l, \cdots, Z_m.@J_m\}$ 和 $S_l \rightarrow S_2 \in (D, \Sigma)^+$，則我們可以得到一個FD，在$S_l \rightarrow S_2$中因為$p_i$移到$q.\tau.\tau_i.@l_i$和$p.@l$、$@J_l, \cdots, Z_m.@J_m$移到$q.\tau.@l$。

3. $\{q, , p_1.\tau.\tau_1 @l_1, \ldots p_n.\tau.\tau_n.@l_n\} \rightarrow q.\tau$ 和 $\{q.\tau, q.\tau.\tau_i.@l_i\} \rightarrow q.\tau.\tau_i$。

5、系統設計

5.1 使用者介面

在 UI 的設計上的考慮，不以 Web Base 為使用者介面，因為在樹狀結構的元件和 Message 即時傳送的處理沒有支援，不方便本系統的開發，故採用 Win From 的操作介面，如圖 5.1

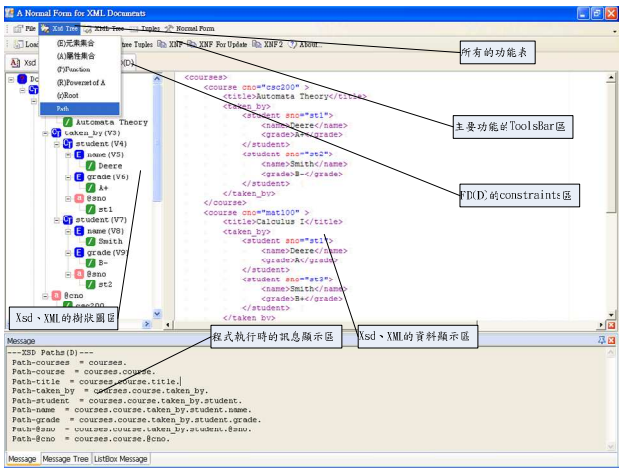

圖 5.1 本系統操作介面

- **所有的功能**:本系統的所有功能選單。
- **主要功能的 ToolBar 區**:把主要的功能放在本區，以方便系統操作如 “Load XML” 、” Load Xsd” 、” Run XNF” 的常用功能。
- **FD(D)的 constraints 區:**主要的功能依據文獻[9]和自定的 FD(D)推演出 XML Schema 的相關 FD(D) constraints。
- **Xsd、XML 的樹狀區:** 本區主要的作用是把 Xsd、XML 檔案 LOAD 到本系統，再解析成一個樹狀結構圖。
- **程式執行時訊息顯示:**本區分成三個部份，分 Message、Tree 、ListBox 三個部份，其中 Message 是顯示程式執行時所產生的訊息，Tree 則把 Tree Tuples 產生的 Tree 顯示在本區，而 ListBox 則是推演出 XML Schema 的相關 FD(D) constraints 產生出來的 FDs(D)放在本區。

5.2 系統架構

我們以 Use Case 圖(圖 5.2)、Class 圖分析系統需求。

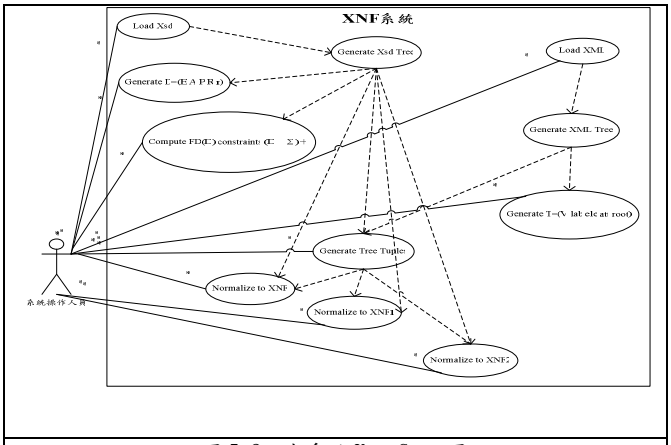

圖 5.2 本系統 Use Case 圖

以下列出分析當中的幾個主要 Use Case

- Generate Xsd Tree(XML Tree)
- Generate D=(E, A, P, R, r)
- Generate T=(V, lab, ele, att, root)
- Compute FD(D) constraints $(D, \Sigma)^+$
- Generate Tree Tuples
- Normalize to XNF
- Normalize to XNF1
- Normalize to XNF2

6、系統實作與實驗結果

我們以 Normalize to XNF2 為例，說明系統之實作。Normalize to XNF2 需使用圖 5.2 三個案例：Xsd Tree、Generate Tree Tuples、Normalize to XNF2。活動執行步驟如圖 6.1 XNF2 活動圖(操作畫面使用的檔案：courses2.Xsd 圖 4.2、courses2.XML 圖 4.3)。

A. 檢查 Tree Tuples 是否存在，如不存在則 Message 區顯示錯誤訊息。

B. 出所有 Tree Tuples，再以 XML Schema 為依據列出所有的 Path(D)，配合使用自定 FD(D)選項，選項為 $\{q, p_1.@l_1, \cdots, p_n.@l_n\} \rightarrow \{z_1.@J_1, \cdots, z_m.@J_m\}$ 型態。

C. 再根據使用者要消除重複資料自定 FD(D)，Creating New Element Types 產生一個新的已消除重複資料的 XML 文件而且此文件再做異動已無需修正其 XML 文件結構和 XML Schema 產生至 Message Tree 區。(圖 6.2 XNF2 操作畫面)。

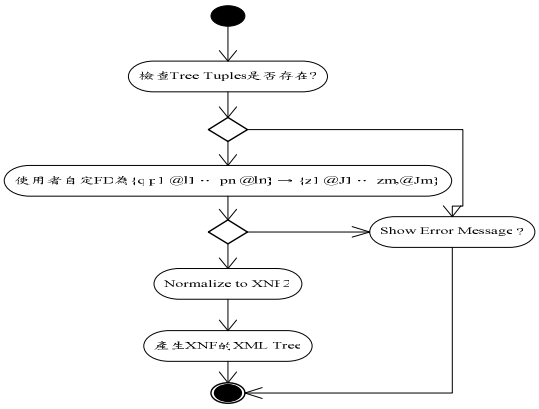

圖 6.1 XNF2 活動圖

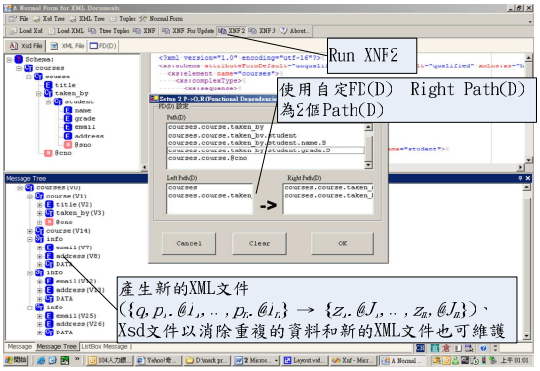

圖 6.2 XNF2 操作畫面

7、結論與未來研究方向

依據 Arenas 與 Libkin 的 XNF 的演算，本論文提出下列幾項修正的作法，並予以實作。有關 XML 文件正規化之未來研究，可以思考下列幾個方向。第一：延續 XFD 及 XML 文件正規化的理論基礎更進一步定義 XMVD，並且提出 XML Tree 上的第四正規化的理論和定義[14]。第二：實作 Web 介面的 XML 正規化工具。

8、參考文獻

[1] K. B. Sall. *XML Family of Specifications : A Practical Guide.* Addison Wesley Profressional, 2002.

[2] M. Arenas. and L. Libkin. "A Normal Form for XML Documents" *ACM Transactions on Database System(TODS),*19(1) pp. 195-232,2004.

[3] M. Arenas, Normalization Theory for XML, SIGMOD Record, Vol. 35, No. 4, December 2006

[4] D. Lee. and W. W. Chu. "CPI : Conparative analysis of six XML schema languages" *ACM SIGMOD Record* ,29(3),pp. 76-87,2000.

[5] XML Query. http://www.w3.org/Query

[6] Michael Benedik , Chee Yong Chan , Wenfei Fan , Juliana Freire , and Rajeev Rastogi, "capturing both Types and Constraints in Data Integration," Proceedings of the 2003 ACM SIGMOD international conference on Management of data.

[7] Peter Buneman , Susan Davidson , Wenfei Fan, Carmem Hara, and WangChiew Tan, "Keys for XML," *Computer Networks,* Volume 39,Issue 5 , August 2002 ,pp. 473-487.

[8] C. Beeri, R. Fagin, and J. H. Howard. "A complete axiomatization for functional and multivalued dependencies in database relations" *In Proceedings of ACM SIGMOD International Conference in Management of Data* , pp.47-61,1977.

[9] Wenfei Fan and Leonid Libkin , "On XML Integrity Condtraints in the Presence of DTDs" *,Journal of ACM(JACM),Volume 49,Issue 3, May 2002*, pp. 268-406

[10] HTML 4.01 Specification. http://www.w3.org/TR/REC-html40/

[11] Extensible Markup Language. http://www.w3.org/TR/REC-xml.

[12] M. L. Lee , T. W. Ling, and W. L. Low , "Dedigning functional dependencies for XML" *,EDBT'2002 Lecture Notes in Computer Science* 2287, pp124-141

[13] David W. Embley, Wai Yin Mok," Developing XML Documents with Guaranteed 'Good' Properties", pp. 426-441, In: Proceedings of the 20th International Conference on Conceptual Modeling, Hideko S. Kunii, Sushil Jajodia, Arne Sølvberg (Ed.), Lecture Notes in Computer Science, Springer-Verlag, Yokohama, Japan, LNCS, Vol. 2224, November 2001.

[14] 林穎聰、廖宜恩， "**XML** 文件的多值相依性與第四正規化"，全國計算機會議(NCS'05)，Dec. 15-16，2005